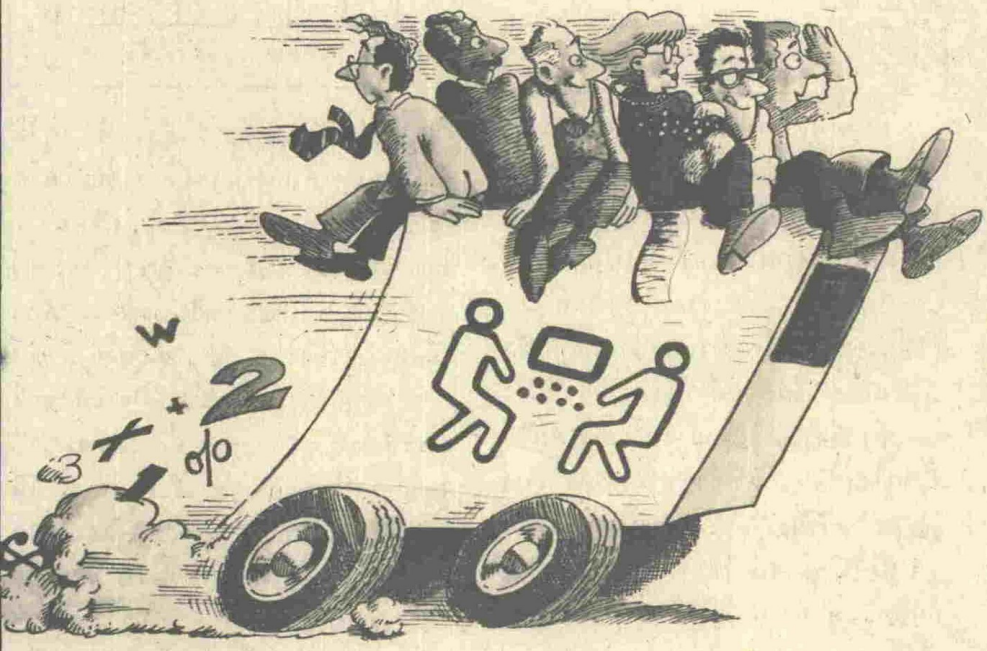
เรื่องน่าอ่าน
ยืน ภู่วรวรรณ
พรพจน์ สุริยะวงศ์

เฉลยข้อสอบ
คอมพิวเตอร์โอลิมปิก



แกะรอยข้อสอบ
คอมพิวเตอร์โอลิมปิก
ที่พรั่งพร้อมไปด้วยเนื้อหา
สาระที่ควรศึกษา

จากที่ผู้เขียนได้ลงบทความเกี่ยวกับการแข่งขันคอมพิวเตอร์โอลิมปิก 34 ณ กรุงเอเธนส์ ประเทศกรีซ ซึ่งผลการแข่งขันครั้งนั้นทีมไทยประสบผลสำเร็จด้วยดี ถึงแม้จะไปแข่งขันครั้งแรกประเทศไทยก็ได้เหรียญเงินหนึ่งเหรียญ และเหรียญทองแดงสองเหรียญ ในฐานะที่ได้รับผิดชอบทางด้านวิชาการและเกี่ยวข้องกับการจัดการฝึกอบรมให้นักเรียนไทยร่วมกับคณาจารย์ต่าง ๆ ได้จัดทำแบบฝึกหัดและโจทย์ต่าง ๆ ไว้เพื่อให้นักเรียนไทยได้มีโอกาสฝึกฝนโจทย์ที่อยู่ในระดับโอลิมปิกจะเป็นโจทย์ที่ค่อนข้างยาก ผู้เขียนจึงได้พยายามหาโจทย์ในระดับนี้ไว้ เพื่อจะได้วิเคราะห์และหาแนวทางในการติวนักเรียนรุ่นใหม่ ๆ ของเราต่อไป

ในวารสารไมโครคอมพิวเตอร์ฉบับที่ 72 ได้นำเอาโจทย์ข้อสอบคอมพิวเตอร์โอลิมปิกที่ใช้สอบจริงสองข้อนำเสนอไปแล้ว ในฉบับนี้ผู้เขียนขอนำเอาเฉลยโจทย์ทั้งสองข้อนี้มาเสนอเพื่อเป็นแนวทางสำหรับผู้ที่สนใจจะศึกษากันต่อไป

โจทย์ข้อ 1

โจทย์ข้อ 1 นี้เป็นการกรอกตัวเลข 1 ถึง 25 ลงในตารางเมตริกซ์ขนาด 5×5 โดยให้ตัวเลขลงในช่องเป็นไปตามกฎ 3 ข้อ คือ เมื่อกรอกช่อง X, Y แล้วช่องต่อไปคือ Z, W

(Z, W) = (X±3, Y)
(Z, W) = (X, Y±3)
(Z, W) = (X±2, Y±2)

โดยการกรอกจะเริ่มกรอก 1 ที่ใดก็ได้ หลังจากนั้นให้คำนวณหารูปแบบที่เป็นไปได้ เมื่อกรอก 1 อยู่ในส่วนบนขวาของ main diagonal รวมช่องที่อยู่บน main diagonal ด้วย โจทย์ข้อนี้ต้องทำตามกฎซึ่งถ้าพิจารณาให้ดีแล้วจะเห็นว่า เมื่อกำหนด X, Y แล้ว ขั้นต่อไปคือ Z, W จะมีทางเป็นไปได้ 8 ทาง แต่บางทางอาจเลยนอกเมตริกซ์ ปัญหานี้จึงเป็นปัญหาแบบเวียนบรรจบ การเฉลยในที่นี้ผู้เขียนขอนำเอาผลของนักเรียนไทยที่ทำได้คะแนนเต็ม 100 คะแนน คือ คุณพิเชษฐ์ ชาวหา มาให้ดู ข้อสอบที่ทำนี้ทำได้รายละเอียดและการแสดงผลเป็นตารางตรงตามวัตถุประสงค์คณะกรรมการทุกประการ โปรแกรมที่เฉลยเป็นดังรูปที่ 1

โจทย์ข้อ 2

โจทย์ข้อสองที่นำเสนอนี้เป็นโจทย์ที่อยู่ในระดับที่ยาก นักเรียนไทยของเราทำไม่ผ่านครบทั้งหมด คุณอัศวินสามารถสร้าง s-term ได้ แต่ยังไม่สามารถทำ reduction และ normalize ได้ คะแนนที่ได้ 40 คะแนน นับว่าอยู่ในเกณฑ์ที่สูงแล้วเพราะถ้าได้คะแนนมากกว่านี้อีกนิดเดียวก็จะได้เหรียญทอง ข้อเฉลยที่นำมาเสนอนี้เขียนโดยคุณพรพจน์ สุริยะวงศ์ โดยใช้ภาษาซี ผู้ที่สนใจอาจนำเอาโปรแกรมนี้ไปเปลี่ยนเป็นปาสคาลได้ เพื่อให้เกิดความเข้าใจในโจทย์ จะขออธิบายโจทย์เพิ่มเติมสักเล็กน้อย โจทย์เริ่มต้นด้วยการกำหนด s-term ใด ๆ ประกอบด้วยตัวอักษร S และสัญลักษณ์วงเล็บ เช่น

```
((((SS) (SS))S) (SS))
```

วงเล็บปิดทางขวาไม่มีความหมาย จึงเขียนได้เป็น

```
((((SS (SSS (SS
```

จากโจทย์ตรงนี้ต้องทำความเข้าใจให้ดีว่า s-term คืออะไร s-term คือ การประกอบของเทอมต่าง ๆ ซึ่งเป็นไปตามกฎที่เขียนด้วย BNF ดังนี้

```
s-term : :- S! (S-term  s-term
```

หรือกล่าวโดยคำพูดได้ว่า s-term ก็คือเทอมที่ประกอบด้วย S หรือ (s-term s-term ซึ่งจุดนี้จะเห็นว่า หากนำ s-term สองตัวมาสร้าง s-term ตัวใหม่ต้องมี (ดังนั้น (จึงเสมือนเป็นโอเปอเรเตอร์ ซึ่งเขียนในรูปฟอร์มแบบ prefix เช่น

```
A+B     กรณีนี้เขียนแบบ infix
AB+     เขียนแบบ postfix
+AB     เขียนแบบ prefix
```

ให้ (เป็นโอเปอเรเตอร์ s-term จึงเขียนในรูปของ prefix นั่นเองหรือกล่าวได้ว่า การเขียน s-term จะเป็นไปตามกฎของ prefix นั่นเอง ซึ่งหมายถึงโอเปอเรเตอร์และโอเปอแรนด์ที่สัมพันธ์กัน

เมื่อเข้าใจแล้วว่า s-term คืออะไร เราก็คงจะอ้างอิงหรือเปรียบเทียบกับ prefix ซึ่งรูปฟอร์มเขียนเหมือนเป็นต้นไม้ เช่น

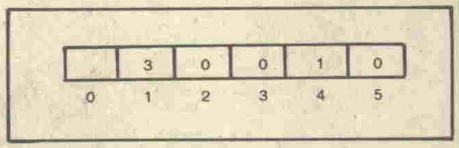
```
                +
+AB เขียนได้เป็น  A     B
```

จากหลักการต้นไม้นี่เอง เราจึงมีวิธีการแทนโครงสร้างข้อมูลของ s-term ได้หลายแบบ แต่แบบที่เฉลยนี้ได้จัดดำเนินการให้โปรแกรมทำงานได้เร็วมาก และมีประสิทธิภาพ ซึ่งต้องอาศัยเทคนิค (วิธีการสร้างหรือตรวจสอบ s-term จึงมีได้หลายวิธีแล้วแต่รูปแบบโครงสร้างข้อมูล) พิจารณาตัวอย่าง s-term ต่อไปนี้ ((SS(SS จะพบว่าจากหลักการของ prefix ทำให้เราได้หลักการบางอย่างขึ้นมา คือใน prefix ใด ๆ จำนวนโอเปอเรเตอร์ต้องน้อยกว่าโอเปอแรนด์อยู่ 1 ซึ่งเราจะเอาข้อนี้มาใช้ เช่น s-term ที่ยกมานี้เป็น s-term ลำดับ 4 จะมีโอเปอเรเตอร์อยู่ 3 ตัว

โจทย์ข้อย่อย 1 บอกให้เราสร้าง s-term ที่ความยาวจาก 1-n โดยแต่ละความยาวต้องสร้าง s-term ที่เป็นไปได้ให้ครบทุกตัว โดยให้โปรแกรมรับค่า n จาก 1 ถึง 10 แล้วสร้างแฟ้มเก็บ s-term แยกเป็นแฟ้ม ๆ การแทนข้อมูลเป็นเรื่องสำคัญ เพราะเรารู้ว่า s-term มีโครงสร้างเป็นต้นไม้แบบ prefix หากแทนแบบต้นไม้ก็ได้ แต่จะมีวิธีการสร้างและตรวจสอบตัวซ้ำยาก เพราะจะต้องใช้เวลามากในการตัดตัวซ้ำออก ในที่นี้ใช้หลักของอะเรย์สำหรับ s-term โดยการแทนค่าดังตัวอย่าง

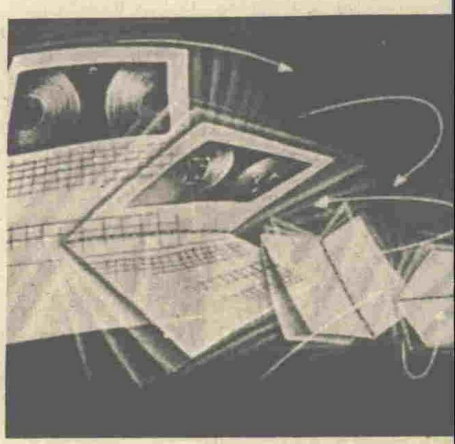
```
s-term  (((SSS(SS
```

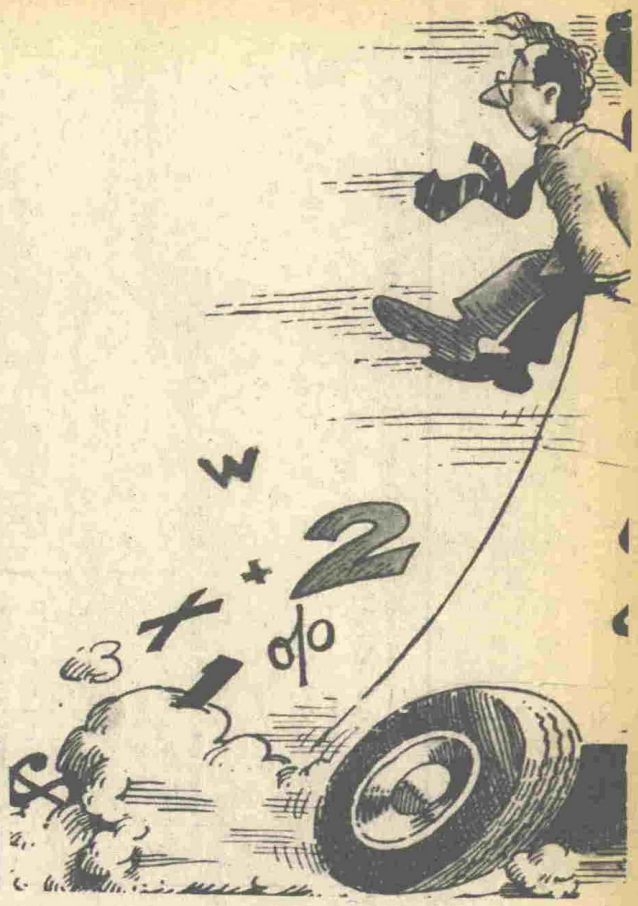
เมื่อแทนจะใช้สล้อตแทน S ดังนั้นจึงมี 5 สล้อต (ในอะเรย์) แต่สำหรับสล้อตที่ 0 จะว่างและไม่ใช้ เพื่อว่าบางภาษาเรียกใช้ไม่ได้ ดังนั้นเมื่อแทนข้อมูลจะได้



สล้อตที่ 1 คือ S ตัวแรกบอกได้ว่ามีโอเปอเรเตอร์นำหน้า 3 ตัว หากไม่มีจะใส่ 0 S ตัวที่ 4 มี (นำหน้า 1 ตัว จากคุณสมบัติของ prefix จะพบว่าจำนวนตัวเลขรวมในอะเรย์จะต้องน้อยกว่าจำนวน S อยู่ 1 ดังรูปที่ 2 เช่น

จากกฎเกณฑ์ของการใส่ตัวเลขลงในสล้อตของ s-term นี้เองจะพบว่ามีกฎที่จะสร้าง s-term ได้จากอะเรย์นี้ กล่าวคือ สล้อตที่อยู่ทางขวาสุดต้องเป็น 0 เสมอ จะเป็นตัวอื่นไม่ได้ ตัวที่อยู่ถัดมาจะเป็น 1 หรือ 0 ก็ได้ ตัวถัดมาจะเป็น 2 หรือ 1 หรือ 0 ได้และเป็นเช่นนี้ตามลำดับ





ด้วยหลักการนี้เราก็ทำการสร้าง s-term ตามโจทย์ที่ต้องการได้โดยง่ายและเร็วมาก การสร้างนี้ทำแบบ recursive ด้วย ลองพิจารณาจากตัวโปรแกรมดังรูปที่ 3

ผลของการรันโปรแกรม s-term ที่สร้างขึ้นนี้จะได้ผลโดยเรียกชื่อ โปรแกรม sterm1 ตามด้วยความยาว s-term ผลจะได้ดังรูปที่ 4

โจทย์ส่วนที่ 2 เป็นการ transformation โดยเริ่มจากการอ่าน s-term ที่เป็นสตริง โดยตรวจสอบการจบสตริงด้วย ; . หรือ null การอ่านนี้ควรตรวจสอบว่าสตริงที่ใส่เข้าไปเป็น s-term หรือไม่ การตรวจสอบนี้ทำได้โดยให้โปรแกรม match กับ (และ S โดยชี้ไปที่สล้อต และถ้าพบ (ก็จะเพิ่มค่า แต่ถ้าเป็น S ก็จะเลื่อนไปสล้อตทางขวาจนกว่าจะหมด ส่วนการ transform จะเข้าใจได้ดีต้องขออธิบายวิธีการตามหลักการ prefix คือ การทำงานตามการคำนวณแบบ prefix เช่น 2*3 เขียนเป็น *23 reduce เป็น 6 นั่นคือ หากพบ (SS จะ reduce เสมือนเป็น s-term ตัวเดียว หรือกล่าวได้ว่า

(SS reduce เป็น s-term

เมื่อเขียนแบบ push down stack ตามหลักการของ automata โดยจะยกจากตัวอย่างดังรูปที่ 5

จะเห็นว่ามีการคำนวณได้ โดยนำการคำนวณด้วยการวิ่งจากซ้ายมาขวาเมื่อเปลี่ยนจากสแตกมาอยู่ในรูปของอะเรย์ที่ใช้แทน ก็ทำได้ง่าย โดยมีขั้นตอนดังนี้

	3	0	0		1	0
(((	S	S	S	(	S	S

เริ่มต้นจากสแกนโดยกำหนดค่าเริ่มต้นให้ n=1 เมื่อพบ (ก็จะเพิ่มค่า 1 ครั้งพบ S ก็จะลดค่า 1 หากผลลัพธ์สุดท้ายเป็น 0 และจบ s-term แล้วจะถือว่า สตริงนั้นเป็น s-term หากไม่ใช่ 0 จะไม่ใช่ s-term การ transformation จึงทำได้ง่าย ตัวอย่างของโปรแกรมแสดงดังรูปที่ 6

เมื่อรันโปรแกรมในส่วนที่สองนี้ ผลของโปรแกรมจะเป็นดังรูปที่ 7

เฉลยส่วนที่ 3 นี้เป็นส่วนที่ยากขึ้น และต้องทำความเข้าใจให้ดีก่อน ส่วนที่สามของโจทย์ให้ทำการ reduce โดยการ reduce เพียงขั้นเดียว (step เดียว) การ reduce จะต้องเป็นไปตามกฎ

(((SA)B)C) reduce ได้เป็น ((AC) (BC))

คราวนี้ลองพิจารณาจากตัวอย่าง ((((SS(SSS(SS ซึ่งแทนข้อมูลเป็น

4 0 1 0 0 1 0

จาก s-rule ซึ่งได้ว่า

(((XA BC จะลดรูปเป็น ((AC(BC

สิ่งที่ต้องทำคือการหา sub s-term A B C ซึ่งจะเห็นได้ว่าจากโครงสร้างอะเรย์เราสามารถแบ่งส่วนของ s-term เป็นตัวย่อย แล้วสร้าง AC, BC กับส่วนหัวและท้ายที่เหลือ จากนั้นนำมาสร้างใหม่เป็น

ส่วนหัว + AC + BC + ส่วนท้าย

การสร้างใหม่นี้ใช้หลักการของการ concat ซึ่งมีหลักการโดยดูจากตัวอย่างรูปที่ 8 เพื่อความเข้าใจดังนี้

ดังนั้นการ concat จึงต้องใส่โอเปอเรเตอร์คือ (นำหน้าไปอีกหนึ่งตัว โปรแกรมทำการ reduce แสดงได้ดังรูปที่ 9

เมื่อรันโปรแกรมและลองใส่ s-term ที่ต้องการ reduce ผลเป็นดังรูปที่ 10

โจทย์ข้อย่อยที่ 4 เป็นการทำการ normalize โดยใช้ s-rule ซ้ำ ๆ ไปจนกว่าจะไม่สามารถ notmalize ได้ หรือถ้าทำครบ 30 ขั้นตอน แล้วยังทำได้ต่ออีกก็บอกว่า normalize ไม่ได้ โปรแกรมส่วนนี้จึงนำจากส่วนข้อ 3 ย่อยมาใช้ดังรูปที่ 11 ผลการรันโปรแกรมส่วนที่ 4 โดยการป๊อป s-term ต่าง ๆ ดูผลได้ดังรูปที่ 12

ส่วนสุดท้าย คือ ส่วนที่ห้าของโจทย์ให้นำโปรแกรมทั้งหมดมารวมกันแล้ว นำ s-term ที่สร้างขึ้นที่มีความยาวต่าง ๆ มาทำการ normalize โดยหาจำนวนตัวที่ normalize ได้กับจำนวนตัวที่ normalize ไม่ได้ โปรแกรมส่วนนี้จะไม่นำเสนอในนี้เพราะจะยาวจึงขอเสนอโปรแกรมในซอฟต์เมล์ผู้สนติดต่อได้ที่วารสารไมโคร ผลของการรันโปรแกรมส่วนสุดท้าย จะเป็นดังรูปที่ 13

เรื่องน่าอ่าน

เฉลยข้อสอบคอมพิวเตอร์โอลิมปิก

รูปที่ 1 เฉลยข้อสอบข้อที่ 1

```
{
Programed by ... Phichet Chaoha, THA2 (Room 5), Thailand.
Language     ... Pascal (Turbo Pascal 5.5)
}

program problem1;

uses    crt;

const   max_n    = 10;  {=== maximum of n ===}
        max_rule = 8;   {=== number of rules ===}

        {=== rules ===}
        rulex:array[1..max_rule] of shortint = (3,-3,0,0,2,2,-2,-2);
        ruley:array[1..max_rule] of shortint = (0,0,3,-3,2,-2,2,-2);

var     n       : byte;  {=== nxn table ===}
        n2      : byte;  {=== square of n ===}
        table   : array[-2..max_n+3,-2..max_n+3] of byte;  {=== the table ===}
        startx,starty : byte;    {=== start (x,y) ===}
        no_solutions  : word;    {=== number of solutions ===}
        step_by_step  : boolean; {=== display mode ===}

(* General proposed routines *)
{=== waitkey ===}
procedure wait;
var ch:char;
begin
    repeat until keypressed;
    ch := readkey;
end;

{=== beep! ===}
procedure beep;
begin
     sound(2000);
     delay(100);
     nosound;
end;
(*--------------------------*)

{=== initialize variables ===}
procedure initialize;
var i,j : shortint;
begin
    {=== initialize table ===}
    for i:=-2 to n+3 do
      for j:=-2 to n+3 do
        if (i<=0) or (i>=n+1) or (j<=0) or (j>=n+1) then
          table[i,j] := 1
        else
          table[i,j] := 0;
end;

{=== make block ===}
procedure makeblock;
const up_right    = '╗';
      up_left     = '╔';
      down_right  = '╝';
      down_left   = '╚';
      vertical1   = '║';
      vertical2   = '╢';
      vertical3   = '╟';
      horizontal1 = '═';
      horizontal2 = '╧';
      horizontal3 = '╤';
      plus        = '╬';

var i,j,k : byte;
begin
     clrscr;
     {=== print horizontal lines ===}
     for i:=1 to n+1 do
     begin
       gotoxy(1,3+2*i);
       for j:=1 to 6*n+1 do
         if (j=1) then write(vertical3)
         else if (j=6*n+1) then write(vertical2)
         else if (i=1) and (j mod 6=1) then write(horizontal3)
         else if (i=n+1) and (j mod 6=1) then write(horizontal2)
         else write(horizontal1);
     end;

     {=== print vertical lines ===}
     for i:=1 to n+1 do
       for j:=1 to n do
       begin
         gotoxy(6*i-5,4+2*j);
         write(vertical1);
       end;

     {=== block decorations ===}
     gotoxy(1,5);          write(up_left);
     gotoxy(6*n+1,5);      write(up_right);
     gotoxy(1,5+2*n);      write(down_left);
     gotoxy(6*n+1,5+2*n); write(down_right);
```

```
      {=== print plus ===}
      for i:=2 to n do
        for j:=2 to n do
        begin
          gotoxy(6*j-5,3+2*i);
          write(plus);
        end;
end;

{=== show output ==}
procedure output;
var i,j : byte;
begin
    {=== show details ===}
    gotoxy(1,1);
    writeln('Start position : ','(',startx,',',starty,') :- ');
    writeln;
    writeln('Solution : ',no_solutions);
    writeln;
   {=== display each number ===}
   for i:=1 to n do
   begin
     for j:=1 to n do
     begin
       gotoxy(6*j-3,4+2*i);
       write(table[i,j]:3);
     end;
     writeln;
   end;

   {=== stop or not? ===}
   if (step_by_step) then
   begin
     writeln;
     writeln;
     writeln('Press any key to continue...');
     wait;
   end;
end;

{=== try to fill number in (x,y) position ===}
procedure fill(x,y,number : byte);
var   tempx,tempy,i,j  : shortint;
      k                : byte;
begin
     {=== fill the (x,y) position ===}
     table[y,x] := number;

     {=== is it all? ===}
     if (number=n2) then
     begin
       inc(no_solutions);
       output;
       exit;
     end;

     {=== not yet then find the next position ===}
     for k:=1 to max_rule do
     begin
       tempx := x+rulex[k];  {=== next x position ===}
       tempy := y+ruley[k];  {=== next y position ===}

       {=== if possible to fill (tempx,tempy) position ===}
       if (table[tempy,tempx]=0) then
       begin
         fill(tempx,tempy,number+1);
         table[tempy,tempx] := 0;
       end;
     end;
end;

{=== input of process1 ===}
procedure input;
var temp : shortint;
begin
     clrscr;
     {=== input n ===}
     repeat
       write('N (5..',max_n,') = '); readln(temp);
     until (temp>=5) and (temp<=max_n);
     n  := temp;
     n2 := n*n;

     {=== input startx ===}
     repeat
       write('Start at X (1..',n,') : '); readln(temp);
     until (temp>0) and (temp<=n);
     startx := temp;

     {=== input starty ===}
     repeat
       write('Start at Y (1..',n,') : '); readln(temp);
     until (temp>0) and (temp<=n);
     starty := temp;
end;

{=== get display mode ===}
procedure getmode;
```

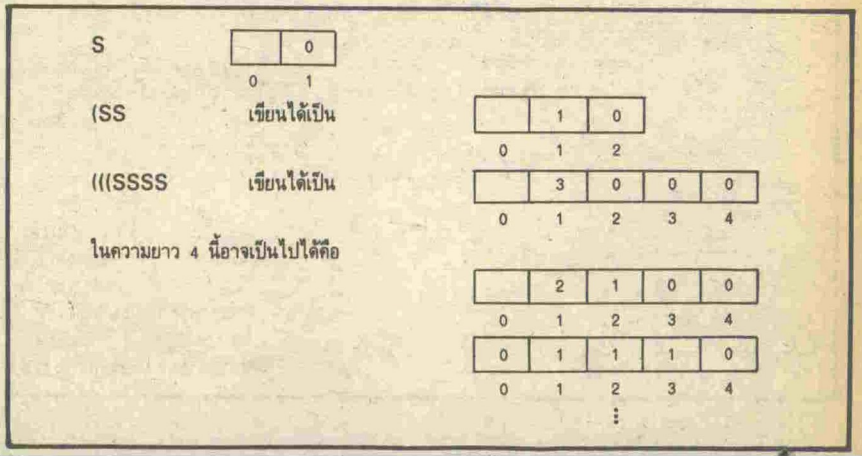
เรื่องน่าอ่าน

เฉลยข้อสอบคอมพิวเตอร์โอลิมปิก

```
var reply : char;
begin
    write('Display step by step or not (Y/N) ?');
    repeat
      repeat until keypressed;
      reply := readkey;
      reply := upcase(reply);
    until (reply in ['Y','N']);
    writeln(reply);
    step_by_step := (reply='Y');
end;

{=== process1 ===}
procedure process1;
begin
     {=== set all variables ===}
     input;
     getmode;
     makeblock;

     {=== initialize variables ===}
     initialize;

     {=== clear number of solutions ===}
     no_solutions := 0;

     {=== try to find all solutions which start at (startx,starty) ===}
     fill(startx,starty,1);

     {=== show number of soltions ===}
     beep;
     writeln;
     writeln('Number of solutions is ',no_solutions,' solutions.');

     {=== wait key to continue ===}
     writeln;
     writeln('Press any key to continue...');
     wait;
end;

{=== process2 ===}
procedure process2;
begin

     {=== clear number of solutions ===}
     makeblock;
     no_solutions := 0;

     {=== try all upper right positions ===}
     for startx:=1 to n do
       for starty:=1 to n do
       if (startx>=starty) then
       begin
         initialize;            {=== initialze table ===}
         fill(startx,starty,1); {=== find solutions ===}
       end;

     {=== show number of solutions ===}
     writeln;
     beep;
     writeln('Number of solutions is ',no_solutions,' solutions and that''s all
');
     wait;
end;

{=== main part ===}
begin

     {=== try to find all solutions which start at (startx,starty) ===}
     process1;

     {=== try to find all solutions which start at the upper right triangle ===

     process2;
end.
```

■

รูปที่ 2



รูปที่ 3 แสดงโปรแกรมที่ทำการสร้าง S-TERM

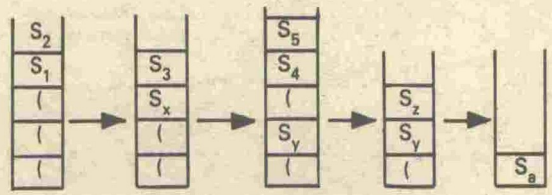
```
/*************************************************/
/*                S-TERM SOLUTION                */
/*                Subproblem no. 1               */
/*             By Pornpote Suriyawong            */
/*                                               */
/* compiler:- microsoft c or turbo c             */
/*************************************************/
#include <stdio.h>

#define MAXSTERM 9999

int StermSlot[MAXSTERM+1];
int OrderOfS;
int nSterm;

FILE *StermFile;
char StermName[20]="Sterm#";

void InitSterm(int order)
{
  int i;

  OrderOfS=order;
  for (i=1;i<=order;i++)
    StermSlot[i]=0;
  StermSlot[1]=order-1;
}
/*************************************************************/
/* ShowSterm writes all s-terms on screen or into file by */
/* selecting the commented portion of procedure.          */
/*************************************************************/
void ShowSterm()
{
  int i,j;

/* writes all s-sterms into a file */

/*  if (nSterm>0) fputs(";\n",StermFile); */
/*  nSterm++;                             */
/*  for (i=1;i<=OrderOfS;i++) {           */
/*    for (j=0;j<StermSlot[i];j++)        */
/*      fputc('(',StermFile);             */
/*    fputc('S',StermFile);               */
/*  }                                     */

/* shows all s-sterms on screen */

  if (nSterm>0) puts(";");
  nSterm++;
  for (i=1;i<=OrderOfS;i++) {
    for (j=0;j<StermSlot[i];j++)
      putchar('(');
    putchar('S');
  }

}

/*********************************************/
/* GenSterm generates a s-sterm recursively */
/*********************************************/
void GenSterm(int nSlot)
{
  /* can not generate sub s-term, prints s-sterm then. */
  if (nSlot<=1) {
    ShowSterm();

  return;
  }
  /* can be generated sub s-term */
  while (StermSlot[1]>=(nSlot-1)) {
    /* generate sub s-term */
    GenSterm(nSlot-1);
    /* adjust '(' in slot */
    StermSlot[nSlot]++;
    StermSlot[1]--;
  }
  /* restore '(' in slot */
  StermSlot[1]+=StermSlot[nSlot];
  StermSlot[nSlot]=0;
}
/*************************************************************************/
/* 1. Write a procedure 'gensterm' to generate s-terms: your procedure */
/*    should generate n (n=length=number of S's) textfiles containing  */
/*    all s-sterms of length 1,...,n repectively. S-terms should ne    */
/*    seperated by ";". The end of the last s-term in each file should */
/*    be marked with ".".                                              */
/*    Write a program that accepts an integer n (<=10), uses the       */
/*    above procedure and displays all generated s-terms on the        */
/*    screen.                                                          */
/*************************************************************************/
void gensterm(int NumOfS)
{
/*  while (NumOfS>0) {                */
/*    itoa(NumOfS,&StermName[6],10);  */
/*    StermFile=fopen(StermName,"w"); */

    InitSterm(NumOfS);
    nSterm=0;
    GenSterm(NumOfS);
    puts(".");

/*    fputs(".\xla",StermFile);       */
/*    fclose(StermFile);              */
/*    NumOfS--;                       */
/*  }                                 */
}

main(int argc,char **argv)
{
  /*********************************************************/
  /* requests the user the order of s-term from command line */
  /*********************************************************/
  if (argc<2) {
    puts("Usage: sterm1 <order_of_Sterm>");
    exit(0);
  }
  /*************************/
  /* then generates s-sterms */
  /*************************/
  gensterm(atoi(argv[1]));
}
```

รูปที่ 4 แสดงผลการรันโปรแกรม STERM 1

```
Lists of outputs of sterm1 1...5

      C>sterm1 1
      S.

      C>sterm1 2
      (SS.

      C>sterm1 3
      ((SSS;
      (S(SS.

      C>sterm1 4
      (((SSSS;
      ((S(SSS;
      (S((SSS;
      ((SS(SS;
      (S(S(SS.

      C>sterm1 5
      ((((SSSSS;
      (((S(SSSS;
      ((S((SSSS;
      (S(((SSSS;
      (((SS(SSS;
      ((S(S(SSS;
      (S((S(SSS;
      ((SS((SSS;
      (S(S((SSS;
      (((SSS(SS;
      ((S(SS(SS;
      (S((SS(SS;
      ((SS(S(SS;
      (S(S(S(SS.
```

รูปที่ 5 ตัวอย่างการเขียน push down stack

(((S1 S2 S3 (S4 S5 มีลักษณะการ reduce อย่างไร



รูปที่ 6 แสดงโปรแกรมที่ทำการ Transformation

```
/*************************************************/
/*                S-TERM SOLUTION                */
/*                Subproblem no. 2               */
/*             By Pornpote Suriyawong            */
/*                                               */
/* compiler:- microsoft c or turbo c             */
/*************************************************/
#include <stdio.h>

#define MAXSTERM 9999

#define EOS       '\0'
#define SEMICOLON ';'
#define PERIOD    '.'
#define ISEOSTERM(c) (c==SEMICOLON || c==PERIOD || c==EOS)

int StermSlot[MAXSTERM+1];
int OrderOfS;
/*********************************************************************/
/* 2. Choose an appropriate data structure for representing s-terms */
/*    which can facilitate the application of the s-rule. Write two */
/*    prodedures 'readterm' and 'printterm' that transform s-terms  */
/*    to (and from) your representation from (and to) the form      */
/*    generated by 'gensterm'. Your program should be able to       */
/*    demonstrated these transformations.                           */
/*********************************************************************/

/*********************************************************************/
/* printterm outputs from data representation to s-term and shows */
/* on computer screen.                                            */
/*********************************************************************/
void printterm()
{
  int i,j;

  for (i=1;i<=OrderOfS;i++) {
    for (j=0;j<StermSlot[i];j++)
      putchar('(');
    putchar('S');
  }
  puts("");
}
/*************************************************************/
/* readterm transforms s-term string into data representation. */
/* It also checks the syntax of the input s-term. It will show */
/* an error message if it finds a wrong s-term, and also shows */
/* a s-term which was transformed.                             */
/*************************************************************/
void readterm(char *StermString)
{
  int n=1;

  OrderOfS=0;
  StermSlot[OrderOfS+1]=0;
  while (!ISEOSTERM(*StermString) && n>0) {
```

```
    if (*StermString=='(') {
      StermSlot[OrderOfS+1]++;
      n++;
    } else if (*StermString=='S') {
      OrderOfS++;
      StermSlot[OrderOfS+1]=0;
      n--;
    } else {
      puts("ERROR: unexpected char found");
      printterm();
      exit(1);
    }
    StermString++;
  }
  if (n!=0 || !ISEOSTERM(*StermString)) {
    puts("ERROR: wrong Sterm");
    printterm();
    exit(1);
  }
}

main(int argc,char **argv)
{
  /*********************************************/
  /* requests user a s-term from command line */
  /*********************************************/
  if (argc<2) {
    puts("Usage: sterm2 <Sterm_Expression>");
    exit(0);
  }
  /*************************************************************/
  /* transforms the input s-term into data-representation */
  /*************************************************************/
  readterm(argv[1]);
  /*********************************************/
  /* retranforms and shows s-term on screen */
  /*********************************************/
  printf("sterm ->");
  printterm();
}
```

รูปที่ 7
แสดงผลโปรแกรม Transformation

```
Lists of outputs of sterm2

      C>sterm2 (S((((SSS((SSS((SSS
      sterm ->(S((((SSS((SSS((SSS

      C>sterm2 (S((((SSS((SSSS((SS
      ERROR: wrong Sterm
      (S((((SSS((SSSS
```

รูปที่ 8 ตัวอย่างการสร้างค่าโดยใช้หลักการ concat

```
(SS                                    (SS
10                                     10
        เมื่อ concat ได้ ((SS(SS หรือ
                  2 0 1 0
```

รูปที่ 9 แสดงโปรแกรมที่ทำการ reduce

```
/*************************************************/
/*                S-TERM SOLUTION                */
/*                Subproblem no. 3               */
/*             By Pornpote Suriyawong            */
/*                                               */
/* compiler:- microsoft c or turbo c             */
/*************************************************/
#include <stdio.h>

#define MAXSTERM 9999
#define SUBSTERM 5999

#define EOS       '\0'
#define SEMICOLON ';'
#define PERIOD    '.'
#define ISEOSTERM(c) (c==SEMICOLON || c==PERIOD || c==EOS)

int StermSlot[MAXSTERM+1];
int OrderOfS;
/*************************************************************/
/* declares some new sub s-sterm buffers AC,BC, and T, and */
/*   also the corresponding order buffers.                   */
/*************************************************************/
int StermAC[SUBSTERM+1],StermBC[SUBSTERM+1],StermT[SUBSTERM+1];
int OrderOfAC,OrderOfBC,OrderOfC,OrderOfT;

/*********************************************************************/
/* bollows two functions 'printterm and readterm' from problem 2 */
/*********************************************************************/


"โปรแกรมส่วนนี้มาจากเฉลยข้อ 2 ย่อยในโจทย์ข้อที่ 2"


/*************************************************************/
/* ParseSterm parses a sub s-sterm by specify a slot number. */
/*   It will return the next slot number which matches the   */
/*   sub s-sterm, or -1 if sub s-sterm is wrong.             */
/*************************************************************/
int ParseSterm(int Slotno)
```

```
{
  int n=0;

  while ((n+=StermSlot[Slotno])!=0) {
    n--;
    if (++Slotno>OrderOfS)
      return -1;
  }
  return Slotno;
}
/*************************************************************/
/* IsNormalized checks if a s-sterm can be normailzed by     */
/*   seeking for every slot which is more than 3. It returns */
/*   the slot number, otherwise return 0.                    */
/*************************************************************/
int IsNormalize()
{
  int i;

  for (i=1;i<=OrderOfS;i++)
    if (StermSlot[i]>=3)
      return i;
  return 0;
}
/*********************************************************************/
/* BreakSterm decomposes a s-sterm at given slot number into         */
/*   sub s-sterm A, B, C, and T. Then it precomposes sub s-sterm     */
/*   AC and BC.                                                      */
/*********************************************************************/
void BreakSterm(int Slotno)
{
  int i,j,k,l;

  j=ParseSterm(Slotno+1);
  OrderOfAC=j-Slotno;
  if (OrderOfAC>=SUBSTERM) {
    puts("PANIC ERROR: SubTermA Overflow");
    exit(1);
  }
  for (i=1;i<=OrderOfAC;i++)
    StermAC[i]=StermSlot[i+Slotno];

  k=ParseSterm(j+1);
  OrderOfBC=k-j;
  if (OrderOfBC>=SUBSTERM) {
    puts("PANIC ERROR: SubTermB Overflow");
    exit(1);
  }
  for (i=1;i<=OrderOfBC;i++)
    StermBC[i]=StermSlot[i+j];

  l=ParseSterm(k+1);
  OrderOfC=l-k;
  if ((OrderOfAC+OrderOfC)>=SUBSTERM || (OrderOfBC+OrderOfC)>=SUBSTERM) {
    puts("PANIC ERROR: SubTermAC or BC Overflow");
    exit(1);
  }
  for (i=1;i<=OrderOfC;i++)
    StermAC[i+OrderOfAC]=StermBC[i+OrderOfBC]=StermSlot[i+k];
  StermAC[1]++;
  StermBC[1]++;

  OrderOfT=OrderOfS-l;
  if (OrderOfT>=SUBSTERM) {
    puts("PANIC ERROR: SubTermT Overflow");
    exit(1);
  }
  for (i=1;i<=OrderOfT;i++)
    StermT[i]=StermSlot[i+l];
}
/*************************************************************/
/* RebuildSterm composes all sub s-sterm into a new s-sterm. */
/*   By concatinating subterm AC, BC, and T to the first part */
/*   of the old s-sterm.                                     */
/*************************************************************/
void RebuildSterm(int Slotno)
{
  int i;

  OrderOfS+=(OrderOfC-1);
  if (OrderOfS>=MAXSTERM) {
    puts("PANIC ERROR: Sterm Overflow");
    exit(1);
  }

  StermSlot[Slotno]-=2;
  StermSlot[Slotno]+=StermAC[1];

  for (i=2,Slotno++;i<=(OrderOfAC+OrderOfC);i++,Slotno++)
    StermSlot[Slotno]=StermAC[i];

  for (i=1;i<=(OrderOfBC+OrderOfC);i++,Slotno++)
    StermSlot[Slotno]=StermBC[i];

  for (i=1;i<=OrderOfT;i++,Slotno++)
    StermSlot[Slotno]=StermT[i];
}
/*********************************************************************/
/* 3. Write a procedure 'reduce' to perform one reduction step       */
/*    according to the s-rule on a specified subterm of an s-term    */
/*    given in your representation. Your program should be able to   */
/*    demonstrate this.                                              */
/*********************************************************************/
void reduce(int Slotno)
{
  /*************************************/
  /* decomposes s-term at given Slotno and */
  /* precomposes into subterm AC, BC and T */
  /*************************************/
  BreakSterm(Slotno);
  /*****************************************/
  /* composes all subterms into a new s-term */
  /*****************************************/
  RebuildSterm(Slotno);
```

```
}

main(int argc,char **argv)
{
  /*****************************************************************/
  /* requests for a given s-term and sub s-term from command line */
  /*****************************************************************/
  if (argc<3) {
    puts("Usage: sterm3 <Sterm_Expression> <Slot_number>");
    exit(0);
  }
  /*************************************/
  /* transforms into your representation */
  /*************************************/
  readterm(argv[1]);
  /*********************************/
  /* outputs the original s-sterm */
  /*********************************/
  printterm();
  /*********************************/
  /* performs one reduction step */
  /*********************************/
  reduce(atoi(argv[2]));
  /*********************************/
  /* outputs the resulting s-sterm */
  /*********************************/
  printf(" ->");
  printterm();
}
```

```
Lists of outputs of sterm3

      C>sterm3 ((((SSSS(((SSSS 1
      ((((SSSS(((SSSS
       ->(((SS(SS(((SSSS

      C>sterm3 ((((SSSS(((SSSS 2
      ((((SSSS(((SSSS
       ->((((SS(((SSSS(S(((SSSS
```

รูปที่ 10 แสดงผลโปรแกรมที่ลองใส่ S-TERM

รูปที่ 11

```
/*************************************************/
/*                S-TERM SOLUTION                */
/*                Subproblem no. 4               */
/*             By Pornpote Suriyawong            */
/*                                               */
/* compiler:- microsoft c or turbo c             */
/*************************************************/
#include <stdio.h>

#define MAXSTERM 9999
#define SUBSTERM 5999
#define TIMECOUNT 30

#define EOS       '\0'
#define SEMICOLON ';'
#define PERIOD    '.'
#define ISEOSTERM(c) (c==SEMICOLON || c==PERIOD || c==EOS)

int StermSlot[MAXSTERM+1];
int OrderOfS;
int StermAC[SUBSTERM+1],StermBC[SUBSTERM+1],StermT[SUBSTERM+1];
int OrderOfAC,OrderOfBC,OrderOfC,OrderOfT;

/*************************************************/
/* bollows some functions from problem 2 and 3 */
/*************************************************/
```

"โปรแกรมในส่วนนี้จะใช้ฟังก์ชัน

Printterm

Readterm

Parsesterm

Isnormalize

Breaksterm

reduce

จากข้อย่อยที่ 2 และ 3 ในโจทย์ข้อที่ 3

```
void reduce(int Slotno)
{
  BreakSterm(Slotno);
  RebuildSterm(Slotno);
  /*****************************************/
  /* modifies to show each step of reduction */
  /*****************************************/
  printf(" -> ");
  printterm();
}
/*********************************************************************/
/* 4. Write a procedure 'normalize' : given an s-term, it should   */
/*    repeatedly choose a subterm to apply the s-rule, until       */
/*    no further reductions are possible or until the number of    */
/*    reduction steps exceeds some maximum, e.g. 30. Your programs */
/*    should be able to demonstrate this.                          */
/*********************************************************************/
int normalize()
{
  int Slotno,time=0;

  while ((Slotno=IsNormalize())!=0 && time<TIMECOUNT) {
    time++;
    reduce(Slotno);
  }
  if (time>=TIMECOUNT)
    return -1;

  return time;
}

main(int argc,char **argv)
{
  /*************************************************/
  /* Request for a given s-term from command line */
  /*************************************************/
  if (argc<2) {
    puts("Usage: sterm4 <Sterm_Expression>");
    exit(0);
  }
  /*************************************************/
  /* transforms s-sterm into your representation */
  /*************************************************/
  readterm(argv[1]);

  /*****************************************/
  /* outputs the s-sterm before normalize */
  /*****************************************/
  printterm();
  /*************************************************************/
  /* performs normalization, outputs the resulting sterm     */
  /* or 'not normalized' message if unsuccessful in 30 steps */
  /*************************************************************/
  if (normalize()>=0)
    printterm();
  else
    puts("not normalized");
}
```

รูปที่ 12 แสดงผลโปรแกรมป๊อป S-TERM

```
Lists of outputs of sterm4 ((((((SSSSSSS

      C>sterm4 ((((((SSSSSSS
      ((((((SSSSSSS
       -> (((((SS(SSSSS
       -> ((((SS((SSSSS
       -> (((SS(((SSSSS
       -> (((SS((SS(SSS
       -> ((SS(((SS(SSS
       -> ((SS((SS((SSS
      ((SS((SS((SSS

Lists of outputs of sterm4 (S((((SSS((SSS((SSS

      C>sterm4 (S((((SSS((SSS((SSS
      (S((((SSS((SSS((SSS
       -> (S(((S((SSS(S((SSS((SSS
       -> (S((((SSS((SSS((S((SSS((SSS
       -> (S(((S((SSS(S((SSS((S((SSS((SSS
       -> (S((((SSS((S((SSS((SSS((S((SSS((S((SSS((SSS
       -> (S(((S((S((SSS((SSS(S((S((SSS((SSS((S((SSS((S((SSS((SSS
          .
          .
          .
       not normalized
```

รูปที่ 13 แสดงผลโปรแกรมที่ทำการ normalize

```
Lists of outputs of sterm5 8

      C>sterm5 8
      ((SS((SS((SS(SS  using 9 steps
      not normalized
      ((S((S((SS((SSSS((S((S((SS((SSSS((SS((S((SS((SSSS  using 11 steps
      ((S((S((S(SSS(S((S(SSS(S((S((S(SSS(S((S(SSS  using 9 steps
      ((S(SS((S(SS((SS(SS  using 7 steps
      ((S((SS((SS(SSS  using 4 steps
      (S((SS((SS((SSS  using 6 steps
      ((SS((SS((SS(SS  using 8 steps
      not normalized
      not normalized
            .
            .
            .
      ((S(S(SS(S(S(SS  using 0 steps
      (S((S(SS(S(S(SS  using 0 steps
      ((SS((SS(S(S(SS  using 0 steps
      (S(S((SS(S(S(SS  using 0 steps
      ((S(S(S(S(SS(S(S(S(S(SS  using 1 steps
      ((S(SS(S(S(S(SS  using 0 steps
      (S((SS(S(S(S(SS  using 0 steps
      ((SS(S(S(S(S(SS  using 0 steps
      (S(S(S(S(S(S(SS  using 0 steps

      NumberOfSterm=429 NotNormalized=41
```